情報デザイン専攻

画像情報処理論及び演習I

-画像合成・類推-

Texture Synthesis/Inpainting

第８回講義
水曜日１限
教室6218情報処理実習室

吉澤　信
shin@riken.jp, 非常勤講師
大妻女子大学　社会情報学部

独立行政法人 理化学研究所

Shin Yoshizawa: shin@riken.jp

レポート1採点結果

1. みなさん、レポート1の採点結果を取りに来てください！
2. 今後、遅れて出しても可(減点0.8倍).
3. やってない問題や間違っていた問題も再提出をがんばってp(^^).
4. △点の人はプログラムに簡単な間違いがあるので、修正して再提出すれば、残りを加点します。例えば△10点ならの15点×0.8=12点加算します.
5. 注意:
   - ✓ ソースコードのコピーは(元の人も)ダメ(0点): 相談や教えるのは可.
   - ✓ 名前と学籍番号なしは0点→再提出してください.

第2回のレポートは来週(6/18)〆切なのでがんばってーp(^^)q

Shin Yoshizawa: shin@riken.jp

今日の授業内容

www.riken.jp/brict/Yoshizawa/Lectures/index.html
www.riken.jp/brict/Yoshizawa/Lectures/Lec08.pdf

① 演習:大津法・第二回レポート.
② Morphing・Texture Synthesis・Inpainting
③ 画像類推.

第2回のレポートは来週(6/18)〆切なのでがんばってーp(^^)q

Shin Yoshizawa: shin@riken.jp

復習:3次元形状を用いた画像合成

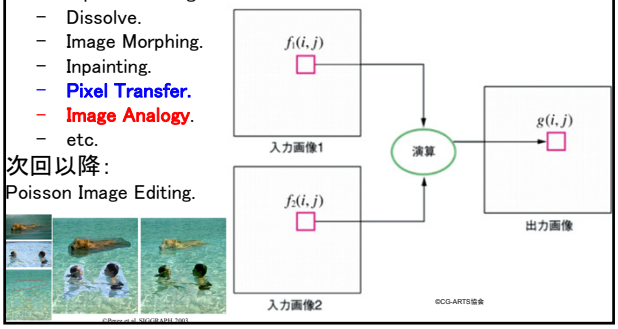
2D人顔・人体画像の3D形状モデルを用いたアニメーション・モーフィング:

Shin Yoshizawa: shin@riken.jp

画像合成(Image Synthesis)

✓ 複数(又は局所画像)画像から新しい画像を生成する事.
- 本講義では3D形状は使わない画像合成を扱う.
- Alpha-Blending.
- Dissolve.
- Image Morphing.
- Inpainting.
- **Pixel Transfer.**
- **Image Analogy.**
- etc.

次回以降:
Poisson Image Editing.



Shin Yoshizawa: shin@riken.jp

単純な合成

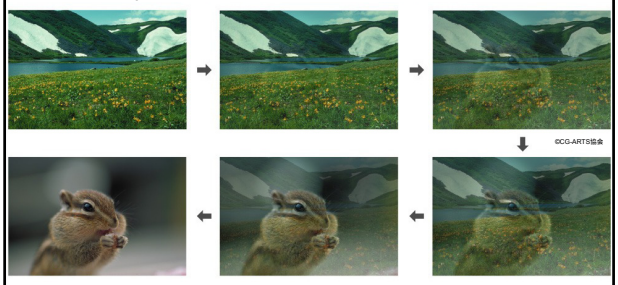
✓ 色の平均: $I^{new}(\mathbf{x}) = (I_1(\mathbf{x}) + I_2(\mathbf{x}))/2$

✓ Alpha-Blending:透明度を画素の位置により線形補間.

Shin Yoshizawa: shin@riken.jp

時間変化の合成：ディゾルブ(Dissolve)

✓ 透明度(Alpha)を時間的に変化(線形補間0.0～1.0)：

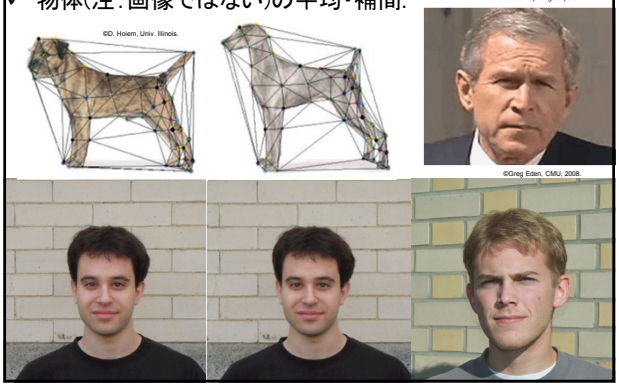


$$A^{\text{new}}(\mathbf{x}) = tA_1(\mathbf{x}) + (1-t)A_2(\mathbf{x}), \quad 0 \le t \le 1.$$

Shin Yoshizawa: shin@riken.jp

モーフィング(Morphing)

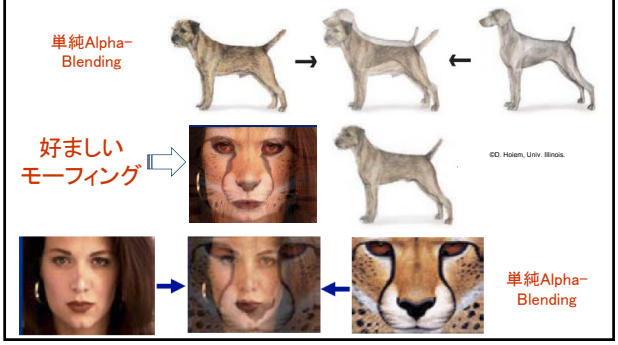
✓ 物体(注：画像ではない)の平均・補間.



Shin Yoshizawa: shin@riken.jp

モーフィング(Morphing)2

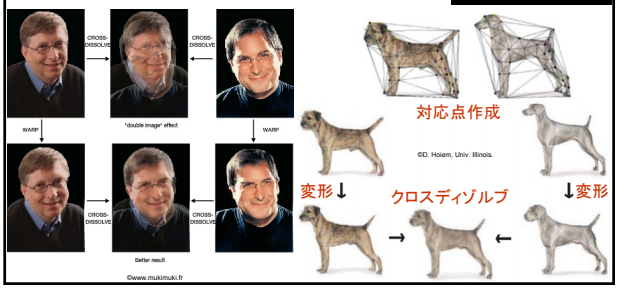
✓ 単純な画素値のディゾルブの結果では物体の平均・補間にはならない！



Shin Yoshizawa: shin@riken.jp

モーフィング(Morphing)2

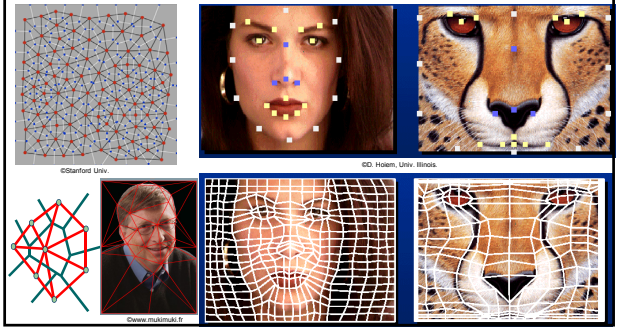
1. 対応点の作成：特徴点作成＋対応付け.
2. 局所変形(Local Warping): 位置合わせ.
3. クロスディゾルブ(Cross-Dissolve).



Shin Yoshizawa: shin@riken.jp

特徴点作成

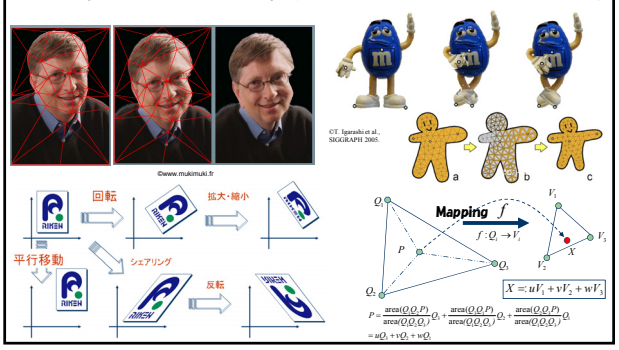
✓ マニュアル、特徴抽出、メッシュ生成(Voronoi図/Delaunay三角形分割)等: Delaunay三角形分割はVoronoi図の双対.



Shin Yoshizawa: shin@riken.jp

局所変形(Local Warping)1

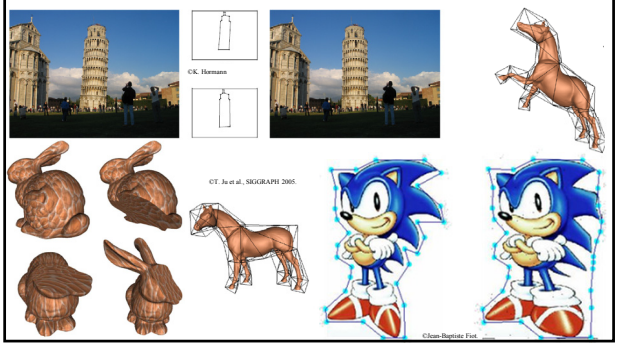
✓ 変形・補間法を用いる：アフィン変換、スプライン補間、重心座標、一般化重心座標、RBF (Radial Basis Function)等.



Shin Yoshizawa: shin@riken.jp

局所変形(Local Warping)2

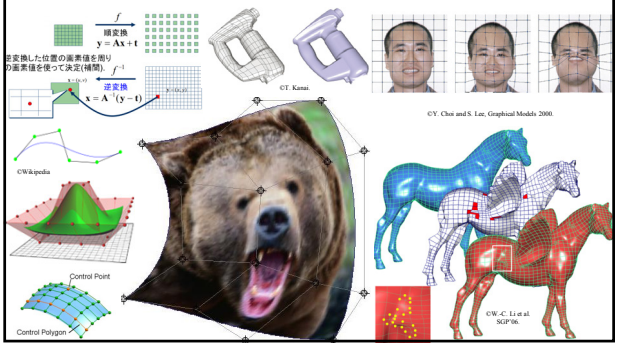
✓ 変形・補間法を用いる：アフィン変換、スプライン補間、重心座標、一般化重心座標、RBF (Radial Basis Function)等.



Shin Yoshizawa: shin@riken.jp

局所変形(Local Warping)3

✓ 変形・補間法を用いる：アフィン変換、スプライン補間、重心座標、一般化重心座標、RBF (Radial Basis Function)等.



Shin Yoshizawa: shin@riken.jp

局所変形(Local Warping)4

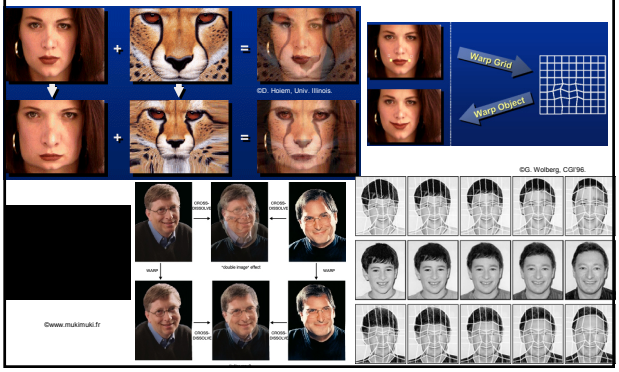
✓ 変形・補間法を用いる：アフィン変換、スプライン補間、重心座標、一般化重心座標、RBF (Radial Basis Function)等.

✓ RBF(沢山の亜種あり)は最もよく用いられているデータ補間法、スプライン補間や重心座標系と比べて格子やメッシュがいらない．写像の裏返りの制御が難しい．

Shin Yoshizawa: shin@riken.jp

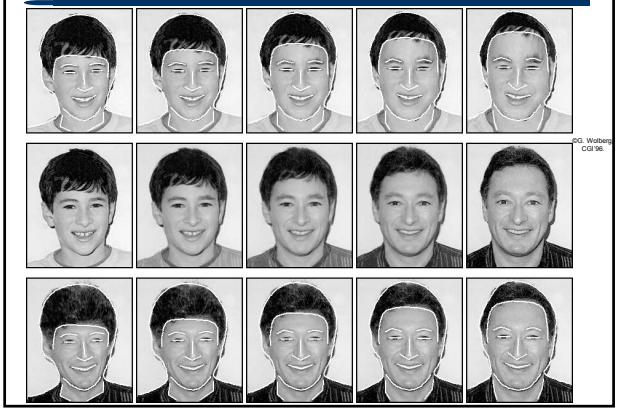
クロスディゾルブ(Cross-Dissolve)1

✓ 複数画像に対する変形結果のディゾルブを計算する事.



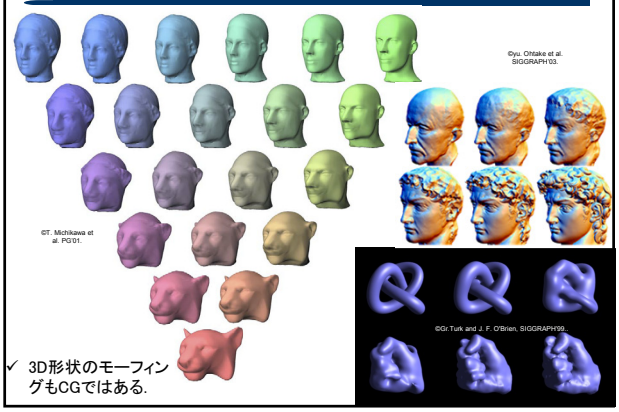
Shin Yoshizawa: shin@riken.jp

クロスディゾルブ(Cross-Dissolve)2



Shin Yoshizawa: shin@riken.jp

モーフィング(Morphing)3D

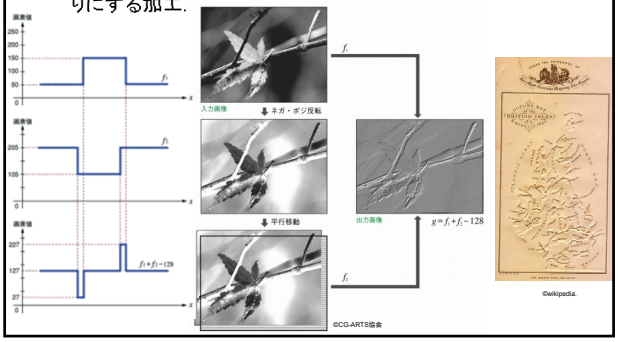


✓ 3D形状のモーフィングもCGではある.

Shin Yoshizawa: shin@riken.jp

フィルタ等の複数の処理を組み合わせる事も

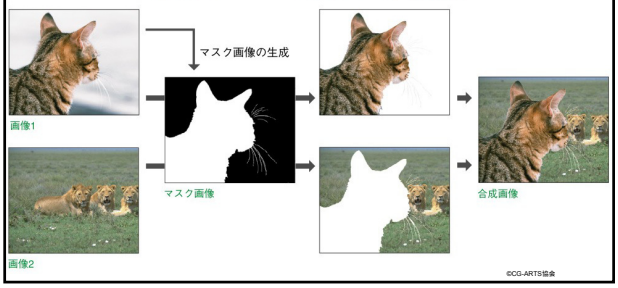
✓ 例えばエンボス画像生成:
- エンボス(Emboss):板金や紙などに文字や絵柄などを浮き彫りにする加工.



Shin Yoshizawa: shin@riken.jp

マスク(領域抽出)画像と画像合成

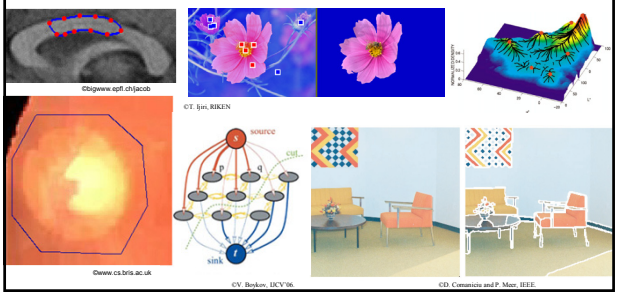
✓ マスク(領域抽出)画像を用いて対象領域だけ合成する事が主流.
- マスクの境界からの距離等を用いる方法もある.
- 領域抽出の応用.



Shin Yoshizawa: shin@riken.jp

マスク(領域抽出)画像生成1

✓ マスク画像は自動領域抽出、クロマキー、マニュアル、半自動(Interactive)等で生成.
- 復習:自動領域抽出:大津の二値化法, Snake (Active Contour), Graph Cuts, Mean Shift, Water Shed (Region Growing)等.



Shin Yoshizawa: shin@riken.jp

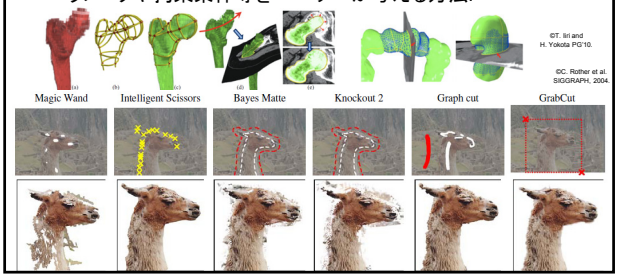
マスク(領域抽出)画像生成2

✓ マスク画像は自動領域抽出、クロマキー、マニュアル、半自動(Interactive)等で生成.
- クロマキー(Chromakey):特定の色からマスクを生成する事.
- テレビ、映画等の背景合成.
- 光学式、回路式、デジタル式.

Shin Yoshizawa: shin@riken.jp

マスク(領域抽出)画像生成3

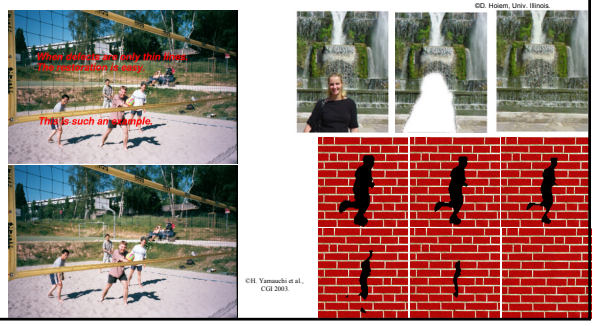
✓ マスク画像は自動領域抽出、クロマキー、マニュアル、半自動(Interactive)等で生成.
- 半自動:最小限のユーザーインタラクションでマスクを生成.
- 基本的アルゴリズムは全自動の領域抽出法だが、抽出法のパラメータや拘束条件等をユーザーが与える方法.



Shin Yoshizawa: shin@riken.jp

Inpainting・Hole Filling

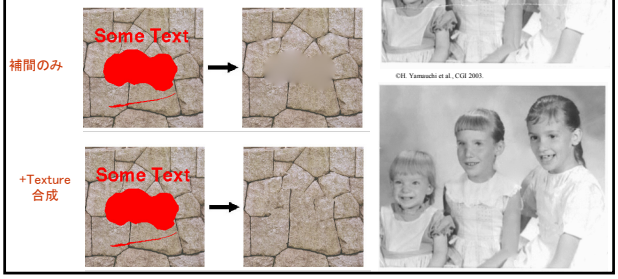
✓ マスク内部の画像を自動生成する事.
- 周りの画素値を使った補間.
- Texture合成: Pixel/Texture Transfer, Image Completion.



Shin Yoshizawa: shin@riken.jp

補間によるInpainting

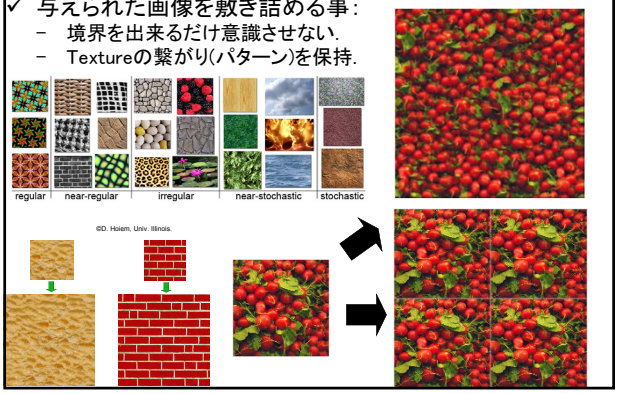
- ✓ 補間によるInpaintingは、(ほとんどの補間法が滑らかな関数で値を繋ぐため)細い領域に有効だが、大きなマスクでは不自然な結果.
- ✓ 通常は補間＋Texture合成.



Shin Yoshizawa: shin@riken.jp

Texture合成

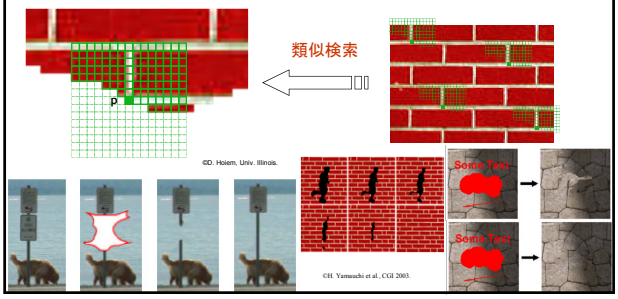
- ✓ 与えられた画像を敷き詰める事:
  - 境界を出来るだけ意識させない.
  - Textureの繋がり(パターン)を保持.



Shin Yoshizawa: shin@riken.jp

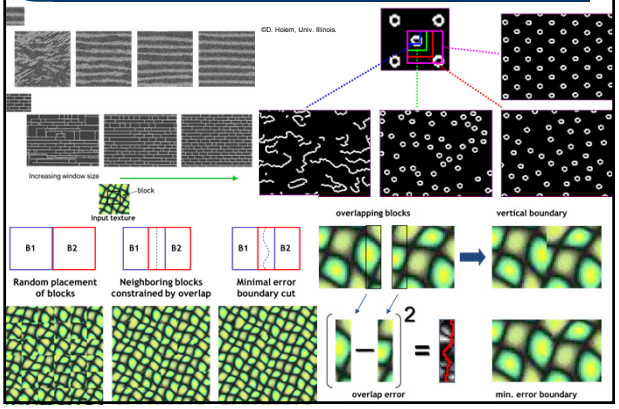
Pixel TransferによるInpainting

- ✓ 画像から似ている画素・Textureを持ってくる.
  - 局所Windowで類似パターンを検索:Windowサイズに依存.
  - 低周波画像は補間で生成しておくと影等の効果を反映出来る.
  - 穴(マスク)を埋める順番が重要！



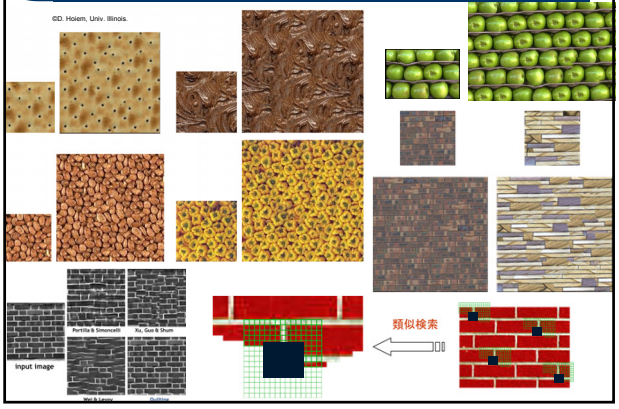
Shin Yoshizawa: shin@riken.jp

Pixel TransferによるInpainting2



Shin Yoshizawa: shin@riken.jp

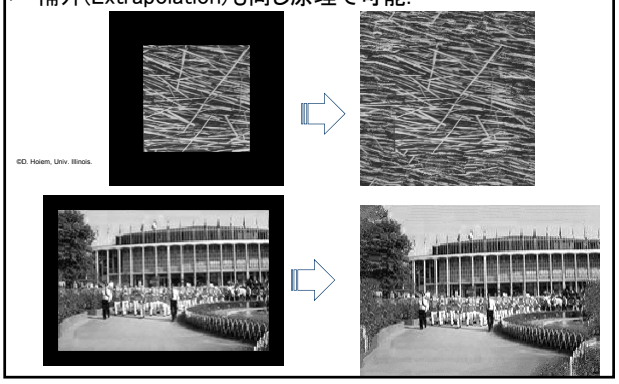
Pixel TransferによるInpainting3



Shin Yoshizawa: shin@riken.jp

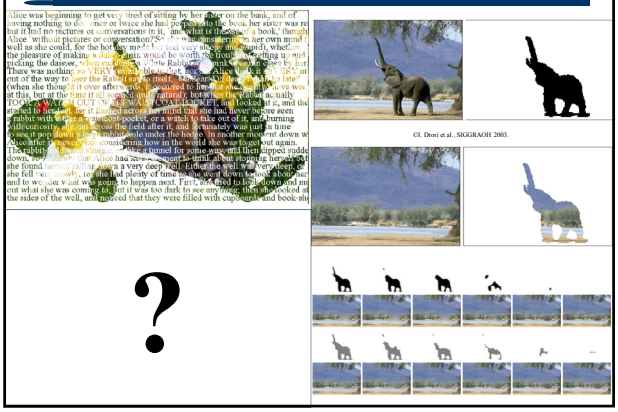
Pixel TransferによるInpainting4

- ✓ 補外(Extrapolation)も同じ原理で可能.



Shin Yoshizawa: shin@riken.jp

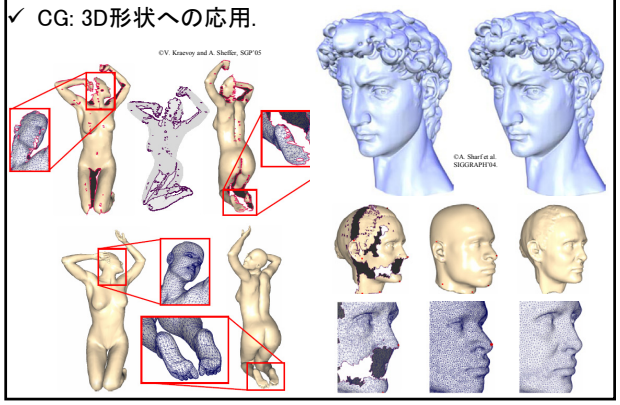
Pixel TransferによるInpainting5



Shin Yoshizawa: shin@riken.jp

Shape Completion

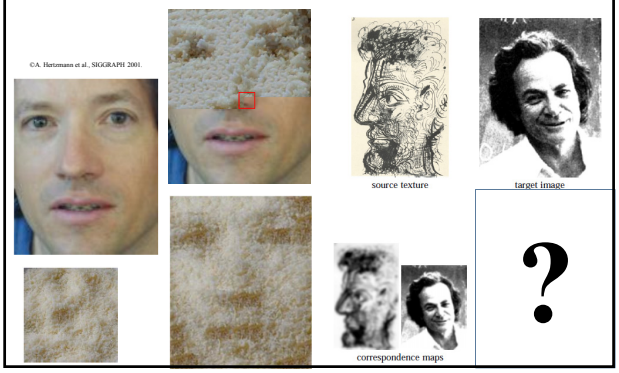
✓ CG: 3D形状への応用.



Shin Yoshizawa: shin@riken.jp

Image Analogy/Example-based

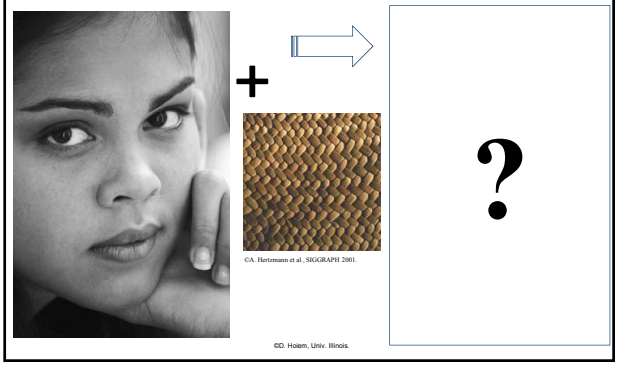
✓ 例題や類推による画像編集・合成:原理は同じ類似検索.



Shin Yoshizawa: shin@riken.jp

Image Analogy2

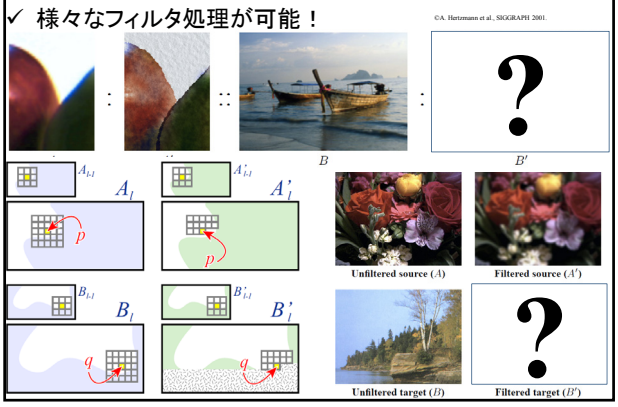
✓ 例題や類推による画像編集・合成:原理は同じ類似検索.



Shin Yoshizawa: shin@riken.jp

Image Analogy3

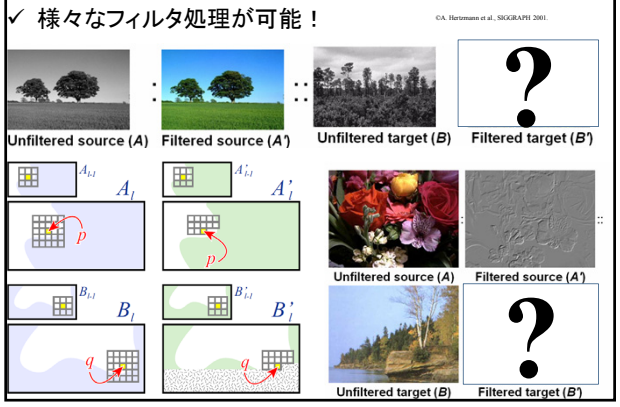
✓ 様々なフィルタ処理が可能！



Shin Yoshizawa: shin@riken.jp

Image Analogy4

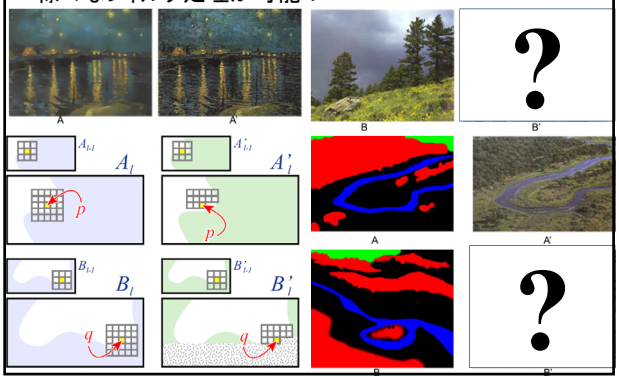
✓ 様々なフィルタ処理が可能！



Shin Yoshizawa: shin@riken.jp

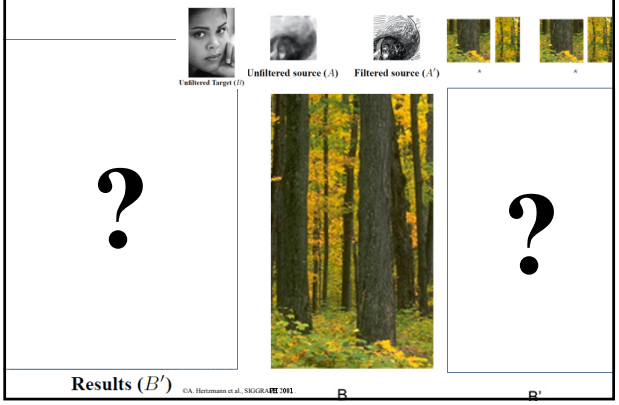
Image Analogy5

✓ 様々なフィルタ処理が可能！



Shin Yoshizawa: shin@riken.jp

Image Analogy6

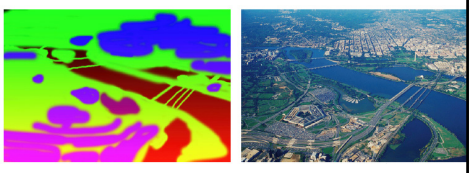


Shin Yoshizawa: shin@riken.jp

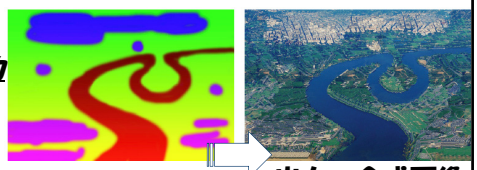
Image Analogy7

Example-based Painting:

データ入力
画像とその領域の分類



Userの入力
Painting

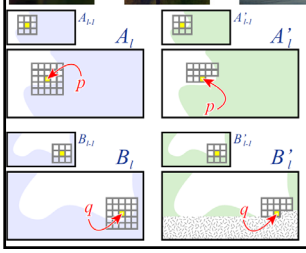


出力：合成画像

Shin Yoshizawa: shin@riken.jp

Image Analogyアルゴリズム

パラメータ(Windowサイズ): r>=2.



CreateImageAnalogy(A, A', B)
1 Compute Gaussian pyramids for (A, A', B)
2 Compute features for (A, A', B)
3 Initialize search structures
4 **for** $\ell \leftarrow 0$ **to** L
5 **for** each pixel $q \in B'_\ell$, in scan-line order
6 $p \leftarrow$ BestMatch$(A, A', B, B', s, \ell, q)$
7 $B'_\ell(q) \leftarrow A'_\ell(p)$
8 $s_\ell(q) \leftarrow p$
9 **return** B'_L

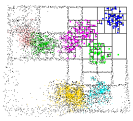
Shin Yoshizawa: shin@riken.jp

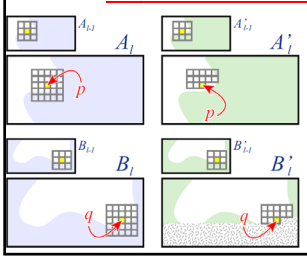
Image Analogyアルゴリズム2

✓ 検索はANN (Approximate Nearest Neighbor)ライブラリを使う.

✓ ANNはエラー(誤差)を許して高速にn次元空間の近傍をサーチ.

パラメータ(ANNError): E>=1.0.





CreateImageAnalogy(A, A', B)
1 Compute Gaussian pyramids for (A, A', B)
2 Compute features for (A, A', B)
3 Initialize search structures
4 **for** $\ell \leftarrow 0$ **to** L
5 **for** each pixel $q \in B'_\ell$, in scan-line order
6 $p \leftarrow$ BestMatch$(A, A', B, B', s, \ell, q)$
7 $B'_\ell(q) \leftarrow A'_\ell(p)$
8 $s_\ell(q) \leftarrow p$
9 **return** B'_L

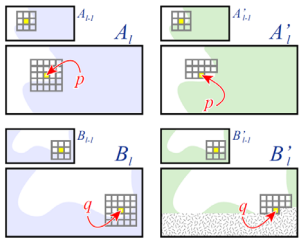
Shin Yoshizawa: shin@riken.jp

Image Analogyアルゴリズム3

✓ Best Approximate Matchは Windowの半径2のとき55次元ベクトルのガウス相関.
- ガウス相関：中心の画素からガウス関数で重みを付けて対応する画素を要素とするベクトルの距離.

✓ Best Coherence Matchは Textureの整合性を加味して既に合成された画素の対応する画素でサーチ.
- Textureの整合性を重視する場合はパラメータkを大きくする.
- 大きくしすぎるとAとA'だけしか結果に反映されないので注意.

パラメータ(Texture度): k>=0.



BestMatch$(A, A', B, B', s, \ell, q)$
1 $p_{app} \leftarrow$ BestApproxMatch(A, A', B, N', ℓ, q)
2 $p_{coh} \leftarrow$ BestCoherenceMatch$(A, A', B, N', s, \ell, q)$
3 $d_{app} \leftarrow \|F_\ell(p_{app}) - F_\ell(q)\|^2$
4 $d_{coh} \leftarrow \|F_\ell(p_{coh}) - F_\ell(q)\|^2$
5 **if** $d_{coh} \leq d_{app}(1 + 2^{\ell - L}\kappa)$
6 **then return** p_{coh}
7 **else return** p_{app}

Shin Yoshizawa: shin@riken.jp

演習：Image Analogyを使ってみよう！

www.riken.jp/brict/Yoshizawa/Lectures/Ex05.zip
www.riken.jp/brict/Yoshizawa/Lectures/Lec08.pdf

Image Analogyでフィルタリング：

1. Ex05内に用意されたプログラム群を動かしてみる.
2. Ex05内の画像を用いてImage Analogyによる色々なフィルタリング処理をしてみる.
3. 新しいフィルタリングを考えてみよう！

この演習は第3回レポートの内容なので
頑張ってくださいねーp(^^)q

Shin Yoshizawa: shin@riken.jp

演習：ANNのコンパイル

www.riken.jp/brict/Yoshizawa/Lectures/Ex05.zip
www.riken.jp/brict/Yoshizawa/Lectures/Lec10.pdf

まずはじめに、ANNをコンパイルする.

1. Ex05.zipを展開する.
2. Ex05内にann_1.1.2.zipがあるのでEx05内で展開する.
3. 端末でEx05/ann_1.1.2に入る、もしもデスクトップに展開していたら、「cd ~/Desktop/Ex05/ann_1.1.2」.
4. コンフィギュレーションを行う4.の後に端末で「sh Make-config」でエンターキー.
5. コンパイルする5.の後に端末で「make linux-g++」と打ち込みエンターキーを押す.Ex05/ann_1.1.2/libの下にlibANN.aが出来れば成功.

Shin Yoshizawa: shin@riken.jp

演習：Ex05内の説明

www.riken.jp/brict/Yoshizawa/Lectures/Ex05.zip
www.riken.jp/brict/Yoshizawa/Lectures/Lec08.pdf

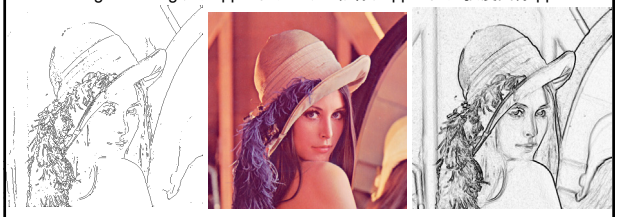
Ex05内の説明：コンパイルは端末で「make」Makefile

- ✓ ImageAnalogyClass.h: Image Analogyの本体.
  - ColorImage.h: カラー画像クラス.
  - GaussianPyramid.h: ガウスピラミッドクラス.
- ✓ Image Analogyとは関係ないファイル：
  - Image Analogyの入力画像を生成するフィルタで使うヘッダーファイル:Gauss.h: ガウス平滑化用、fastgb.h & gaussfgt1D.h:高速エッジ保存フィルター用.
  - 前回までに使ったファイル:SimpleImage.h(画像クラス)、otsu.h(大津の二値化)、ppmio.h(カラー画像入出力)、thinning.h(細線化).

Shin Yoshizawa: shin@riken.jp

演習：Image Analogyとは関係ないファイル

- ✓ まずは、Image Analogyとは直接関係ないプログラムから. ただし、これらのプログラムを使えばImage Analogyに入力させる画像を簡単に作成可能: Run_Smoothing.sh, Run_EdgePreserving.sh, Run_EdgeThinning.sh.
- ✓ EdgeThinning.cxx: エッジ強度画像(勾配強度＝Gradientベクトルの大きさ)とエッジの細線化画像を出力するプログラム:引数3:
  - EdgeThinning 入力.ppm 出力エッジ細線化.ppm 出力強度画像.ppm

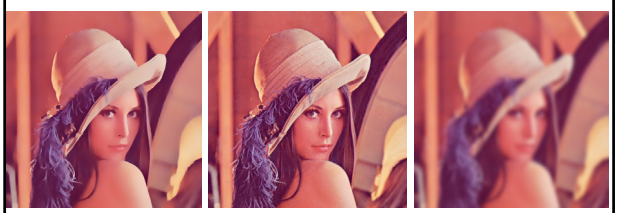


エッジ細線化画像　入力　エッジ強度画像

Shin Yoshizawa: shin@riken.jp

演習：Image Analogyとは関係ないファイル

- ✓ Smoothing.cxx:ガウス平滑化を実行するプログラム:引数3:
  - Smoothing 入力.ppm 出力.ppm 平滑化度合(double)
  - 平滑化度合のパラメータは0より大きな実数2.0～20.0ぐらいが実用的.
- ✓ EdgePreservingFilter.cxx: エッジ保存平滑化を実行: 引数3
  - EdgePreservingFilter 入力.ppm 出力.ppm エッジの大きさ(double)
  - エッジの大きさパラメータは0より大きな実数0.5～2.0ぐらいが実用的.

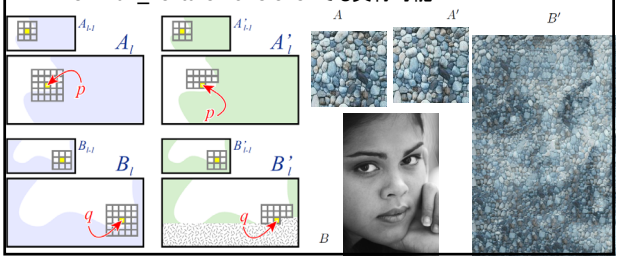


EdgePreservingFilter, 1.0　入力　Smoothing, 5.0

Shin Yoshizawa: shin@riken.jp

演習：Texture Transfer

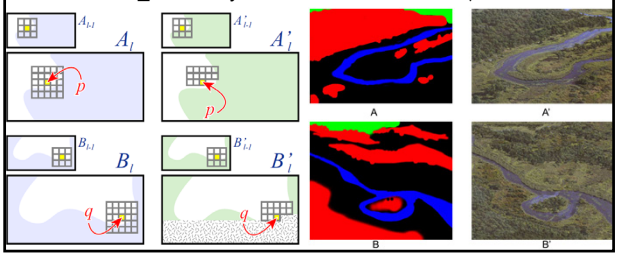
- ✓ Image Analogyを用いてTexture Transferを実行するプログラム: 引数 8
  - TextureTransfer 入力画像A.ppm 入力画像A'.ppm 入力画像B.ppm 出力画像B'.ppm Texture度k(double>=0.0) ANN誤差(double>=1.0) Window半径(int>=2) Blending(1.0>=double>=0.0, 小→元画像強め)
  - sh Run_TextureTransfer.shでも実行可能.



Shin Yoshizawa: shin@riken.jp

演習：Texture by Numbers

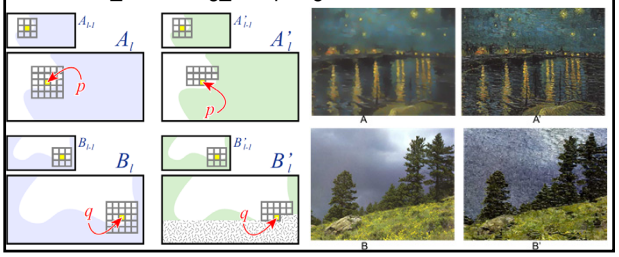
- ✓ Image Analogyを用いてTexture by Numbersを実行するプログラム: 引数 7
  - TextureByNumbers 入力画像A.ppm 入力画像A'.ppm 入力画像B.ppm 出力画像B'.ppm Texture度k(double>=0.0) ANN誤差(double>=1.0) Window半径(int>=2)
  - sh Run_TextureByNumbers.shでも実行可能↓実行結果.



Shin Yoshizawa: shin@riken.jp

演習：Artistic Filters

- ✓ Image Analogyを用いて様々なArtisticフィルタを実行: 引数 7
  - ArtisticFilter 入力画像A.ppm 入力画像A'.ppm 入力画像B.ppm 出力画像B'.ppm Texture度k(double>=0.0) ANN誤差(double>=1.0) Window半径(int>=2)
  - sh Run_ArtisticFilter.sh、sh Run_Etc1～3.sh、sh Run_Smoothing_Sharpning.shでも実行可能.



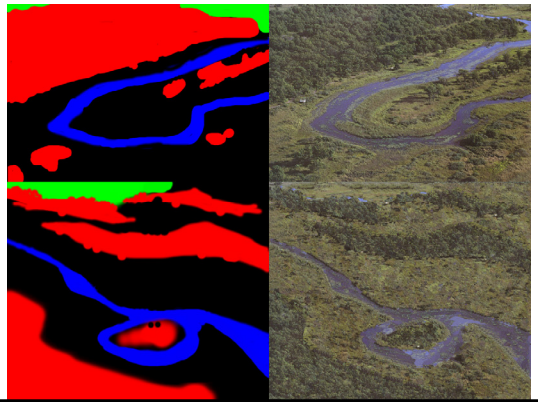
Shin Yoshizawa: shin@riken.jp

演習：シェルの説明

- ✓ 端末にて「sh シェルスクリプト名.sh」で実行、中にコンパイル＋実行＋表示のコマンドが書いてある.
  - Run_TextureTransfer.sh: 5種類のテクスチャーをテクスチャー度を1,3,5,7,9の五種類で実行.

Shin Yoshizawa: shin@riken.jp

演習：Run_TextureByNumbers.sh

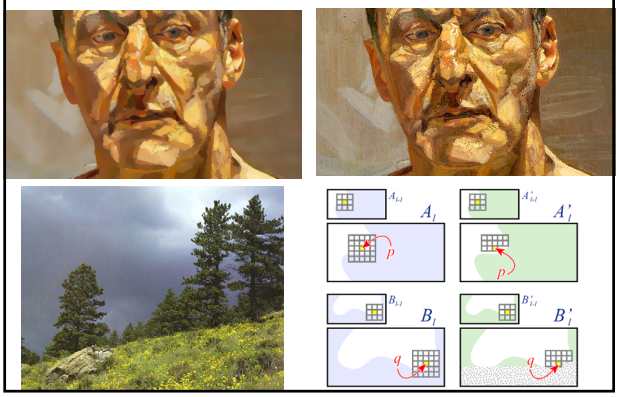


Shin Yoshizawa: shin@riken.jp

演習：Run_Smoothing_Sharpning.sh

Shin Yoshizawa: shin@riken.jp

演習：Run_ArtisticFilter.sh



Shin Yoshizawa: shin@riken.jp

演習:Run_ArtisticFilter.sh

✓ テクスチャー度の違い:6種類実行.

Shin Yoshizawa: shin@riken.jp

演習:Run_ArtisticFilter.sh

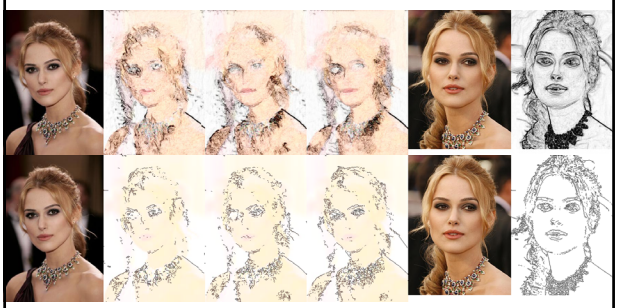
✓ Window半径の違い:2種類実行.

注意点:ANN誤差は全て1000.0で実行、1.0に近ければ綺麗な結果だが、計算時間が大: 数十分～数十時間かかる可能性あり!

Shin Yoshizawa: shin@riken.jp

演習:Run_Etc1.sh

✓ ArtisticFilterにEdgeThinningの出力(エッジ強度)を使った結果.



Shin Yoshizawa: shin@riken.jp

演習:Run_Etc2.sh

Shin Yoshizawa: shin@riken.jp

演習:Run_Etc3.sh

✓ 油絵的フィルタ効果

入力→

Shin Yoshizawa: shin@riken.jp

演習:Run_Etc3.sh

✓ 水彩画的フィルタ効果

入力→

Shin Yoshizawa: shin@riken.jp

演習：シェルスクリプトを動かしてみよう！

端末にて「sh シェルスクリプト名.sh」

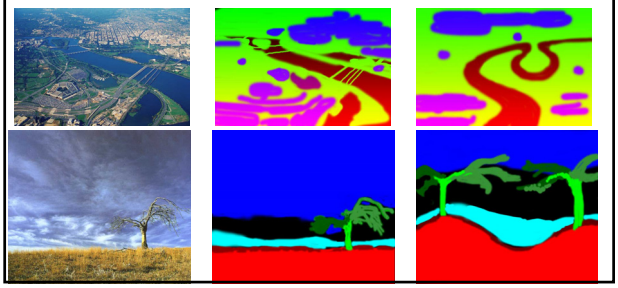
- ✓ Run_ArtisticFilter.sh
- ✓ Run_EdgePreserving.sh
- ✓ Run_EdgeThinning.sh
- ✓ Run_Smoothing.sh
- ✓ Run_Smoothing_Sharpning.sh
- ✓ Run_TextureTransfer.sh
- ✓ Run_TextureByNumbers.sh
- ✓ Run_Etc1.sh
- ✓ Run_Etc2.sh
- ✓ Run_Etc3.sh

Shin Yoshizawa: shin@riken.jp

演習：シェルスクリプトを変えてみよう！

Run_TextureByNumbers.shを使って以下の画像に対して処理してみよう!

A　　A'　　B

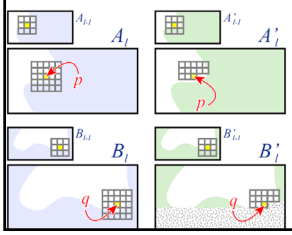


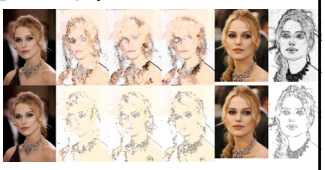
Shin Yoshizawa: shin@riken.jp

演習：自分で新しいエフェクトを作ってみよう！

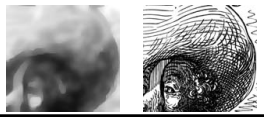
Run_ArtisticFilter.shを使って以下の画像の様に自分のオリジナルのエフェクトを処理してみよう!

注意点：A, A'の画像サイズは同じでないとダメ！





ヒント：　模様・エフェクトが付いた画像を平滑化するとよい？



Shin Yoshizawa: shin@riken.jp

来週の予定

www.riken.jp/brict/Yoshizawa/Lectures/index.html

©Perez et al. SIGGRAPH 2003.

① 画像合成・Inpaintingその2

② 演習: 画像類推・合成.